

# On fast Fourier solvers for the tensor product high-order FEM for a generalized Poisson equation

Alexander Zlotnik <sup>1</sup> and Ilya Zlotnik <sup>2</sup>

## Abstract

We present direct logarithmically optimal in theory and fast in practice algorithms to implement the tensor product high order finite element method on multi-dimensional rectangular parallelepipeds for solving PDEs of the Poisson kind. They are based on the well-known Fourier approaches. The key new points are the fast direct and inverse FFT-based algorithms for expansion in eigenvectors of the 1D eigenvalue problems for the high order FEM. The algorithms can further be used for numerous applications, in particular, to implement the tensor product high order finite element methods for various time-dependent PDEs. Results of numerical experiments in 2D and 3D cases are presented.

**Keywords.** Fast direct algorithm, high order finite element method, FFT, Poisson equation.

**MSC subject classifications.** 65F05, 65F15, 65M60, 65T99.

## 1 Introduction

We present direct fast algorithms to implement  $n$ th order ( $n \geq 2$ ) finite element method (FEM) on rectangular parallelepipeds [4] for solving a  $N$ -dimensional generalized Poisson equation ( $N \geq 2$ ) with the Dirichlet boundary condition. The algorithms are based on the well-known Fourier approaches, e.g., see [2, 10–12] and references therein. The key new points are the fast direct and inverse algorithms for expansion in eigenvectors of the 1D eigenvalue problems for the high order FEM utilizing several versions of the discrete fast Fourier transform (FFT) [3]. This solves the old known problem, see [2, p. 271], and makes the full algorithms logarithmically optimal with respect to the number of elements as in the case of the bilinear elements ( $n = 1$ ) or standard finite-difference schemes. The algorithms are fast in practice (faster than the theoretical expectations) and demonstrate only a mild growth in  $n$  starting from the standard case  $n = 1$ . For example, in the 9th order case, the 2D FEM system for  $2^{20}$  elements containing almost  $85 \cdot 10^6$  unknowns and the 3D FEM system for  $2^{18}$  elements containing more than  $190 \cdot 10^6$  unknowns are solved respectively in less than 2 and 15 min on an ordinary laptop using Matlab R2016a code, see details below.

The algorithms can further serve for a variety of applications including general 2nd order elliptic equations (as preconditioners), for the  $N$ -dimensional heat, wave or time-dependent Schrödinger PDEs, etc. They can be applied for some non-rectangular domains, in particular, by involving meshes topologically equivalent to rectangular ones [7]. Other standard boundary conditions can be covered as well, see a brief description in [14]. Moreover, the Fourier structure of algorithms is especially valuable for solving some wave physics problems, in particular, involving non-local boundary conditions, e.g., see [2, 6, 13], whence our own interest arose. Clearly the algorithms are also highly parallelizable.

---

<sup>1</sup>Corresponding author. Department of Mathematics at Faculty of Economic Sciences, National Research University Higher School of Economics, Myasnitskaya 20, 101000 Moscow, Russia. E-mail: azlotnik2008@gmail.com

<sup>2</sup>Settlement Depository Company, 2-oi Verkhonii Mikhailovskii proezd 9, building 2, 115419 Moscow, Russia. E-mail: ilya.zlotnik@gmail.com

The paper is organized as follows. In Section 2, the statement of the 1D FEM eigenvalue problem together with auxiliary FEM eigenvalue problems on and inside the reference element are given. The basic Section 3 is devoted to a description of eigenvalues and eigenvectors of the 1D FEM eigenvalue problem and the fast direct and inverse algorithms for expansion in these eigenvectors. Applications to the generalized Poisson equation in a  $N$ -dimensional rectangular parallelepiped with the Dirichlet boundary condition are described in Section 4. Results of numerical experiments for  $N = 2$  and  $3$  are presented in detail in Section 5; all of them include the standard case  $n = 1$  for comparison.

## 2 The statement of 1D FEM eigenvalue problem

We first consider in detail the FEM for the simplest 1D eigenvalue ODE problem

$$-u''(x) = \lambda u(x) \quad \text{on } [0, X], \quad u(0) = u(X) = 0, \quad u(x) \not\equiv 0. \quad (2.1)$$

We take the uniform mesh  $\bar{\omega}_h$  with the nodes  $x_j = jh$ ,  $j = \overline{0, K}$  (i.e.,  $0 \leq j \leq K$ ) and the step  $h = X/K$ . Let  $H_h^{(n)}[0, X]$  be the FEM space of the piecewise-polynomial functions  $\varphi \in C[0, X]$  such that  $\varphi(x) \in \mathcal{P}_n$  for  $x \in [x_{j-1}, x_j]$ ,  $j = \overline{1, K}$ , with  $\varphi(0) = \varphi(X) = 0$ ; here  $\mathcal{P}_n$  is the space of polynomials having at most  $n$ th degree,  $n \geq 2$ .

Let  $S_K^{(n)}$  be the space of vector functions  $w$  such that  $w_j \in \mathbb{R}$  for  $j = \overline{0, K}$  with  $w_0 = w_K = 0$  and  $w_{j-1/2} \in \mathbb{R}^{n-1}$ ,  $j = \overline{1, K}$ . Clearly  $\dim S_K^{(n)} = nK - 1$ . A function  $\varphi \in H_h^{(n)}[0, X]$  is uniquely defined by its values at the mesh nodes  $\varphi_j = \varphi(x_j)$ ,  $j = \overline{0, K}$ , with  $\varphi_0 = \varphi_K = 0$ , and inside the elements  $\varphi_{j-1/2} = \{\varphi(x_{j-1} + (l/n)h)\}_{l=1}^{n-1}$ ,  $j = \overline{1, K}$ , that form the element in  $S_K^{(n)}$ .

We use the following scaled operator form of the standard FEM discretization for problem (2.1)

$$\mathcal{A}v = \lambda \mathcal{C}v, \quad v \in S_K^{(n)}, \quad v \neq 0. \quad (2.2)$$

Here  $\mathcal{A} = \mathcal{A}^T > 0$  and  $\mathcal{C} = \mathcal{C}^T > 0$  are the global (scaled) stiffness and mass operators (matrices) acting in  $S_K^{(n)}$  and together with  $\lambda$  independent on  $h$ ; the true approximate eigenvalues are  $\lambda_h = 4h^{-2}\lambda$ .

Let  $A = \{A_{kl}\}_{k,l=0}^n$  and  $C = \{C_{kl}\}_{k,l=0}^n$  be the local stiffness and mass matrices related to the reference element  $\sigma_0 = [-1, 1]$  with the following entries

$$A_{kl} = \int_{\sigma_0} e'_k(x) e'_l(x) dx, \quad C_{kl} = \int_{\sigma_0} e_k(x) e_l(x) dx,$$

where  $\{e_l\}_{l=0}^n$  is the Lagrange basis in  $\mathcal{P}_n$  such that  $e_l(-1 + (2k)/n) = \delta_{kl}$ , for  $k, l = \overline{0, n}$ , and  $\delta_{kl}$  is the Kronecker delta. The matrices  $A$ ,  $C$  and the related matrix pencil  $G(\lambda) := A - \lambda C$  have the following  $3 \times 3$ -block form

$$A = \begin{pmatrix} a_0 & a^T & a_n \\ a & \tilde{A} & \tilde{a} \\ a_n & \tilde{a}^T & a_0 \end{pmatrix}, \quad C = \begin{pmatrix} c_0 & c^T & c_n \\ c & \tilde{C} & \tilde{c} \\ c_n & \tilde{c}^T & c_0 \end{pmatrix}, \quad G(\lambda) = \begin{pmatrix} g_0(\lambda) & g^T(\lambda) & g_n(\lambda) \\ g(\lambda) & \tilde{G}(\lambda) & \tilde{g}(\lambda) \\ g_n(\lambda) & \tilde{g}^T(\lambda) & g_0(\lambda) \end{pmatrix}. \quad (2.3)$$

Here  $\tilde{A}$ ,  $\tilde{C}$  and  $\tilde{G}(\lambda) = \tilde{A} - \lambda \tilde{C}$  are square matrices of order  $n - 1$  with the column vectors  $a, c, g(\lambda) = a - \lambda c \in \mathbb{R}^{n-1}$  whereas  $\tilde{p}_l \equiv (Pp)_l = p_{n-l}$ ,  $l = \overline{1, n-1}$ , for  $p \in \mathbb{R}^{n-1}$ . The matrices  $A$ ,  $C$  and  $G(\lambda)$  are *bisymmetric* (i.e. symmetric with respect to the main and secondary diagonals).

Notice that  $P_{ij} = \delta_{i(n-j)}$  and  $P^T = P^{-1} = P$ . Let  $\mathbb{R}_e^{n-1}$  and  $\mathbb{R}_o^{n-1}$  be the subspaces of even and odd vectors in  $\mathbb{R}^{n-1}$ , i.e. such that respectively  $Pp = p$  and  $Pp = -p$ . The decomposition  $\mathbb{R}^{n-1} = \mathbb{R}_e^{n-1} \oplus \mathbb{R}_o^{n-1}$  (for  $n \geq 3$ ) is implemented by the formulas

$$p = p_e + p_o, \quad p_e := 0.5(p + \check{p}), \quad p_o := 0.5(p - \check{p}). \quad (2.4)$$

Notice that  $\dim \mathbb{R}_e^{n-1} = [n/2]$  and  $\dim \mathbb{R}_o^{n-1} = [(n-1)/2]$ , with  $\mathbb{R}_o^{n-1} = \{0\}$  for  $n = 2$ . Clearly

$$\check{p} \cdot q = p \cdot \check{q}, \quad \check{p} \cdot \check{q} = p \cdot q \quad \text{for any } p, q \in \mathbb{R}^{n-1}. \quad (2.5)$$

Hereafter the symbol  $\cdot$  denotes the inner product of vectors in  $\mathbb{R}^{n-1}$ .

Then problem (2.2) can be written in the following explicit form

$$g_n(\lambda)v_{j-1} + \check{g}(\lambda) \cdot v_{j-1/2} + 2g_0(\lambda)v_j + g(\lambda) \cdot v_{j+1/2} + g_n(\lambda)v_{j+1} = 0, \quad j = \overline{1, K-1}, \quad (2.6)$$

$$g(\lambda)v_{j-1} + \tilde{G}(\lambda)v_{j-1/2} + \check{g}(\lambda)v_j = 0, \quad j = \overline{1, K}, \quad (2.7)$$

with  $v_0 = v_K = 0$ ,  $v \neq 0$ .

We also consider the auxiliary eigenvalue problems on and inside the reference element  $\sigma_0$

$$Ae = \lambda Ce, \quad e \in \mathbb{R}^{n+1}, \quad e \neq 0; \quad (2.8)$$

$$\tilde{A}e = \lambda \tilde{C}e, \quad e \in \mathbb{R}^{n-1}, \quad e \neq 0, \quad (2.9)$$

where clearly  $A \geq 0$ ,  $C > 0$  and  $\tilde{A} = \tilde{A}^T > 0$ ,  $\tilde{C} = \tilde{C}^T > 0$ ; see some their properties in [13] (where the problem similar to (2.6), (2.7) on the uniform mesh on  $[0, \infty)$  for  $\lambda \in \mathbb{C}$  was studied). Denote by  $S_n$  and  $\tilde{S}_n$  their spectra. Let  $\{\lambda_0^{(l)}, e^{(l)}\}_{l=1}^{n-1}$  be eigenpairs of problem (2.9).

**Lemma 2.1.** 1. The subspaces  $\mathbb{R}_e^{n-1}$  and  $\mathbb{R}_o^{n-1}$  are invariant with respect to  $\tilde{A}$  and  $\tilde{C}$ . Thus each eigenvector in  $\{e^{(l)}\}_{l=1}^{n-1}$  can be chosen either even or odd. Also  $\lambda_0^{(l)} > 0$ ,  $l = \overline{1, n-1}$ .

2. Similar properties are valid for problem (2.8) with the exception of one simple zero eigenvalue.

*Proof.* Any bisymmetric matrix  $B$  of the order  $n-1$  commutes with  $P$ , i.e.

$$BP = PB, \quad (2.10)$$

that implies the main result of Item 1. The property  $\lambda_0^{(l)} > 0$ ,  $l = \overline{1, n-1}$  is well-known.

For Item 2, the argument is similar taking into account that  $A(1 \dots 1)^T = 0$  (concerning simplicity of  $\lambda = 0$ , see Proposition 5 in [13]).  $\square$

One can check by the direct computation that all the eigenvalues in  $S_n$  and  $\tilde{S}_n$  are simple and  $S_n \cap \tilde{S}_n = \emptyset$  at least for  $1 \leq n \leq 9$ , see [13].

For low  $n$ , one can find  $S_n$  and  $\tilde{S}_n$  analytically (with the help of Mathematica), in particular,  $\tilde{S}_2 = \{2.5\}$ ,  $\tilde{S}_3 = \{2.5, 10.5\}$ ,  $\tilde{S}_4 = \{14 \pm \sqrt{133}, 10.5\}$  and  $\tilde{S}_5 = \{14 \pm \sqrt{133}, 30 \pm 9\sqrt{5}\}$  (repeatability of the eigenvalues is not occasional, see [13]).

We choose  $\{e^{(l)}\}_{l=1}^{n-1}$  as in Lemma 2.1 using scaling  $\tilde{C}e^{(l)} \cdot e^{(l)} = 1$ .

**Lemma 2.2.** Let  $\tilde{G}(\lambda)p = -g(\lambda)$ , where  $\lambda \notin \tilde{S}_n$ . Let the vectors  $a$  and  $c$  be expanded as

$$a = \sum_{l=1}^{n-1} a^{(l)} \tilde{C}e^{(l)}, \quad c = \sum_{l=1}^{n-1} c^{(l)} \tilde{C}e^{(l)}, \quad \text{with } a^{(l)} = a \cdot e^{(l)}, \quad c^{(l)} = c \cdot e^{(l)}. \quad (2.11)$$

See  $\tilde{G}(\lambda)$ ,  $g(\lambda)$ ,  $a$  and  $c$  in (2.3). Then the following formulas hold

$$p = \sum_{l=1}^{n-1} \frac{a^{(l)} - \lambda c^{(l)}}{\lambda - \lambda_0^{(l)}} e^{(l)} = \sum_{l=1}^{n-1} \frac{a^{(l)} - \lambda_0^{(l)} c^{(l)}}{\lambda - \lambda_0^{(l)}} e^{(l)} - \tilde{C}^{-1}c.$$

*Proof.* For the expansions  $p = \sum_{l=1}^{n-1} p_l e^{(l)}$  and (2.11), we have

$$\tilde{G}(\lambda)p = \sum_{l=1}^{n-1} (\lambda_0^{(l)} - \lambda) p_l \tilde{C} e^{(l)}, \quad g(\lambda) = \sum_{l=1}^{n-1} (a^{(l)} - \lambda c^{(l)}) \tilde{C} e^{(l)},$$

and the result easily follows.  $\square$

### 3 Solving of the 1D FEM eigenvalue problem and related FFT-based algorithms

Below we impose the following assumption:

$$(A) \quad \text{the eigenvalues in } S_n \text{ and } \tilde{S}_n \text{ are simple and } S_n \cap \tilde{S}_n = \emptyset.$$

Recall that it is valid at least for  $2 \leq n \leq 9$  (below in Section 5 we verify it up to  $n = 21$ ).

We introduce the auxiliary equation

$$\hat{\gamma}(\lambda) \equiv -(g_0 - g \cdot \tilde{G}^{-1}g)(\lambda)/(g_n - \check{g} \cdot \tilde{G}^{-1}g)(\lambda) = \theta, \quad (3.1)$$

where  $\lambda \notin \tilde{S}_n$ , with the parameter  $\theta$ . Its solving is equivalent to finding the roots of a polynomial having at most  $n$ th degree, see [13]. In particular, owing to Lemma 2.2 this equation can be rewritten as

$$a_0 - \lambda c_0 + \sum_{l=1}^{n-1} \frac{(a^{(l)} - \lambda c^{(l)})^2}{\lambda - \lambda_0^{(l)}} = -\theta \left( a_n - \lambda c_n + \sum_{l=1}^{n-1} \frac{(\tilde{a}^{(l)} - \lambda \tilde{c}^{(l)})(a^{(l)} - \lambda c^{(l)})}{\lambda - \lambda_0^{(l)}} \right). \quad (3.2)$$

Here  $\tilde{a}^{(l)} = \check{a} \cdot e^{(l)}$  and  $\tilde{c}^{(l)} = \check{c} \cdot e^{(l)}$ . Moreover, for  $2 \leq n \leq 9$  computations help to confirm that the vectors  $e^{(l)}$  are even and odd respectively for odd and even  $l$  provided that  $\lambda_0^{(1)} < \dots < \lambda_0^{(n-1)}$ ; therefore  $\tilde{a}^{(l)} = (-1)^l a^{(l)}$  and  $\tilde{c}^{(l)} = (-1)^l c^{(l)}$ ,  $l = \overline{1, n-1}$ .

We define the simplest inner product in  $S_K^{(n)}$  and the corresponding squared  $\mathcal{C}$ -norm

$$(y, v)_{S_K^{(n)}} := \sum_{j=1}^{K-1} y_j v_j + \sum_{j=1}^K y_{j-1/2} \cdot v_{j-1/2}, \quad \|v\|_{\mathcal{C}}^2 := (\mathcal{C}v, v)_{S_K^{(n)}}.$$

Next theorem describes eigenvalues and eigenvectors of problem (2.2).

**Theorem 3.1.** 1. The spectrum of problem (2.2) consists in  $\tilde{S}_n$  and the numbers  $\{\lambda_k^{(l)}\}_{l=1}^n$  that are all  $n$  (and all positive real) solutions to equation (3.2) with  $\theta = \theta_k := \cos \frac{\pi k}{K}$  for  $k = \overline{1, K-1}$ . The numbers  $\{\lambda_k^{(l)}\}_{l=1}^n$  differ from  $\{\lambda_0^{(l)}\}_{l=1}^{n-1}$  and are different for fixed  $k$ .

2. The following eigenvector corresponds to the eigenvalue  $\lambda_0^{(l)}$ :

$$s_{0,j}^{(l)} = 0, \quad j = \overline{1, K-1}, \quad s_{0,j-1/2}^{(l)} = (-P)^{j-1} e^{(l)}, \quad j = \overline{1, K}, \quad (3.3)$$

for  $l = \overline{1, n-1}$ . Here  $(-P)^{j-1} e = (-1)^{j-1} e$  for even  $e$ ,  $(-P)^{j-1} e = e$  for odd  $e$ .

3. The following eigenvector corresponds to the eigenvalue  $\lambda_k^{(l)}$ :

$$s_{k,j}^{(l)} = s_{k,j}, \quad j = \overline{1, K-1}, \quad s_{k,j-1/2}^{(l)} = p_k^{(l)} \sin \frac{\pi k(j-1)}{K} + \check{p}_k^{(l)} \sin \frac{\pi k j}{K}, \quad j = \overline{1, K}, \quad (3.4)$$

where  $s_{k,j} := \sin \frac{\pi k j}{K}$  and  $p_k^{(l)} \in \mathbb{R}^{n-1}$  solves the non-degenerate algebraic system  $\tilde{G}(\lambda_k^{(l)}) p_k^{(l)} = -g(\lambda_k^{(l)})$ , for  $k = \overline{1, K-1}$ ,  $l = \overline{1, n}$ .

4. The introduced eigenvectors are  $\mathcal{C}$ -orthogonal, i.e.

$$(\mathcal{C} s_k^{(l)}, s_{\tilde{k}}^{(\tilde{l})})_{S_K^{(n)}} = 0 \quad (3.5)$$

for any  $k, \tilde{k} \in \overline{0, K-1}$ ,  $l \in \overline{1, n - \delta_{k0}}$  and  $\tilde{l} \in \overline{1, n - \delta_{\tilde{k}0}}$  such that  $k \neq \tilde{k}$  and/or  $l \neq \tilde{l}$ .

Consequently they form the basis in  $S_K^{(n)}$ , i.e. any  $w \in S_K^{(n)}$  can be uniquely expanded as

$$w = \sum_{l=1}^{n-1} w_{0l} s_0^{(l)} + \sum_{k=1}^{K-1} \sum_{l=1}^n w_{kl} s_k^{(l)}. \quad (3.6)$$

*Proof.* 1. We distinguish between two cases. Let first  $\lambda \in \tilde{S}_n$  and  $e$  satisfy (2.9). Then, for any  $j = \overline{1, K}$ , using equation (2.7) we get

$$0 = v_{j-1/2} \cdot \tilde{G}(\lambda) e = \tilde{G}(\lambda) v_{j-1/2} \cdot e = -[(g(\lambda) \cdot e) v_{j-1} + (\check{g}(\lambda) \cdot e) v_j].$$

Clearly

$$G(\lambda) \begin{pmatrix} 0 \\ e \\ 0 \end{pmatrix} = \begin{pmatrix} g(\lambda) \cdot e \\ 0 \\ \check{g}(\lambda) \cdot e \end{pmatrix}.$$

Since  $\lambda \notin S_n$  by assumption (A), we have  $g(\lambda) \cdot e \neq 0$  or  $\check{g}(\lambda) \cdot e \neq 0$ . Owing to assumption (A) and Lemma 2.1,  $\lambda$  is simple in  $\tilde{S}_n$  and  $e$  is either even or odd. Correspondingly either  $\check{g}(\lambda) \cdot e = g(\lambda) \cdot e \neq 0$  and  $v_j = -v_{j-1}$ , or  $\check{g}(\lambda) \cdot e = -g(\lambda) \cdot e \neq 0$  and  $v_j = v_{j-1}$ . Since  $v_0 = 0$ , in both cases we get

$$v_j = 0, \quad j = \overline{0, K}. \quad (3.7)$$

Thus equation (2.7) is reduced to  $\tilde{G} v_{j-1/2} = 0$  and implies that  $v_{j-1/2} = c_{j-1/2} e$ .

Now equation (2.6) is reduced to

$$c_{j-1/2} \check{g}(\lambda) \cdot e + c_{j+1/2} g(\lambda) \cdot e = 0, \quad j = \overline{1, K-1}.$$

Therefore  $c_{j+1/2} = -c_{j-1/2}$  for even  $e$  or  $c_{j+1/2} = c_{j-1/2}$  for odd  $e$ . Consequently the sought eigenvector  $v$  satisfies (3.7) together with

$$v_{j-1/2} = (-1)^{j-1} e \quad \text{for even } e, \quad v_{j-1/2} = e \quad \text{for odd } e, \quad j = \overline{1, K}$$

( $v$  is defined up to a non-zero constant multiplier). Thus we come to eigenvectors (3.3).

2. Next let  $\lambda \notin \tilde{S}_n$ . Then from equation (2.7) we get

$$v_{j-1/2} = -G^{-1}(\lambda) [v_{j-1} g(\lambda) + v_j \check{g}(\lambda)], \quad j = \overline{1, K}. \quad (3.8)$$

Inserting this into equation (2.6), we find the three-point equation

$$\hat{g}_n(\lambda) v_{j-1} + 2\hat{g}_0(\lambda) v_j + \hat{g}_n(\lambda) v_{j+1} = 0, \quad j = \overline{1, K-1}, \quad (3.9)$$

where

$$\hat{g}_0(\lambda) = (g_0 - g \cdot \tilde{G}^{-1} g)(\lambda), \quad \hat{g}_n(\lambda) = (g_n - \check{g} \cdot \tilde{G}^{-1} g)(\lambda).$$

Straightforwardly the following equalities hold

$$G(\lambda) \begin{pmatrix} 1 \\ -(\tilde{G}^{-1}g)(\lambda) \\ 0 \end{pmatrix} = \begin{pmatrix} \hat{g}_0(\lambda) \\ 0 \\ \hat{g}_n(\lambda) \end{pmatrix}, \quad G(\lambda) \begin{pmatrix} 0 \\ -(\tilde{G}^{-1}\check{g})(\lambda) \\ 1 \end{pmatrix} = \begin{pmatrix} \hat{g}_n(\lambda) \\ 0 \\ \hat{g}_0(\lambda) \end{pmatrix}.$$

If  $\hat{g}_n(\lambda) = \hat{g}_0(\lambda) = 0$ , then the equalities mean that  $\lambda$  is at least double eigenvalue for problem (2.8) that contradicts assumption (A).

If  $\hat{g}_n(\lambda) = 0$  and  $\hat{g}_0(\lambda) \neq 0$ , then equation (3.9) together with (3.8) lead to  $v = 0$  thus such  $\lambda$  does not satisfy (2.2).

Therefore  $\hat{g}_n(\lambda) \neq 0$  and equation (3.9) is simplified to

$$v_{j-1} - 2\hat{\gamma}(\lambda)v_j + v_{j+1} = 0, \quad j = \overline{1, K-1}, \quad (3.10)$$

with the function  $\hat{\gamma}(\lambda) = \hat{g}_0(\lambda)/\hat{g}_n(\lambda)$  (see it also in (3.1)). Since  $v_0 = v_n = 0$ , we can use the expansion

$$v_j = \sum_{k=1}^{K-1} \tilde{v}_k s_{k,j}, \quad j = \overline{0, K},$$

and define the vector  $\tilde{\mathbf{v}} := (\tilde{v}_1, \dots, \tilde{v}_{K-1})$  of its coefficients. Using the expansion in (3.10) gives

$$2 \sum_{k=1}^{K-1} \tilde{v}_k (\theta_k - \hat{\gamma}(\lambda)) s_{k,j} = 0, \quad j = \overline{1, K-1}. \quad (3.11)$$

Clearly this equality is valid for some  $\tilde{\mathbf{v}} \neq 0$  if and only if

$$\hat{\gamma}(\lambda) = \theta_k \quad \text{for some } k = \overline{1, K-1}. \quad (3.12)$$

Notice that  $\tilde{\mathbf{v}} = 0$  is equivalent to  $v = 0$  in  $S_K^{(n)}$  (taking into account formula (3.8)). Therefore  $\lambda$  satisfies (3.12); moreover,  $\tilde{v}_j = \delta_{kj}$  and consequently  $v_j = s_{k,j}$ ,  $j = \overline{0, K}$ , together with

$$v_{j-1/2} = -s_{k,j-1}(G^{-1}g)(\lambda) - s_{k,j}(G^{-1}\check{g})(\lambda), \quad j = \overline{1, K},$$

see (3.8) (all last three equalities are valid up to the same non-zero multiplier). Thus we come to eigenvectors (3.4).

The total amount of eigenvalues  $\lambda \notin \tilde{S}_n$  (taking into account their possible multiplicity) is  $\dim S_K^{(n)} - (n-1) = n(K-1)$ . The maximal amount of roots algebraic equations (3.12) for all  $k$  is the same so that each equation (3.12) has to possess exactly  $n$  distinct roots (for fixed  $k$ , the written eigenvector  $v$  is defined by  $\lambda$  uniquely).

3. Property (3.5) is knowingly valid for eigenvectors  $s_k^{(l)}$  and  $s_{\tilde{k}}^{(\tilde{l})}$  corresponding to different eigenvalues of problem (2.2), in particular, for  $k = 0$  and  $\tilde{k} \neq 0$ , or  $k = \tilde{k}$  and  $l \neq \tilde{l}$ . The remaining case will be covered below in Corollary 3.1 of the related Lemma 3.1.  $\square$

Notice that: (1) the vectors  $s_0^{(l)}$  are used only to describe the algorithm, and only the vectors  $e^{(l)}$  are applied in its implementation; (2)  $s_{k,j}^{(l)}$  are independent on  $l$ ; (3) the vectors  $p_k^{(l)}$  are independent on  $j$  and can also be computed owing to Lemma 2.2.

**Lemma 3.1.** Let  $w \in S_K^{(n)}$  and  $w_{j-1/2} = qw_{j-1} + \check{q}w_j$ ,  $j = \overline{1, K}$ , for some  $q \in \mathbb{R}^{n-1}$ . Then

$$(\mathcal{C}s_0^{(l)}, w)_{S_K^{(n)}} = 0, \quad l = \overline{1, n-1}. \quad (3.13)$$

$$\begin{aligned} & (\mathcal{C}s_k^{(l)}, w)_{S_K^{(n)}} \\ &= 2\{c_0 + c \cdot p_k^{(l)} + (\tilde{C}p_k^{(l)} + c) \cdot q + \theta_k[c_n + c \cdot \check{p}_k^{(l)} + (\tilde{C}\check{p}_k^{(l)} + c) \cdot \check{q}]\}(s_k, w)_{\omega_h}, \end{aligned} \quad (3.14)$$

for  $k = \overline{1, K-1}$ ,  $l = \overline{1, n}$ , where  $(s_k, w)_{\omega_h} := \sum_{j=1}^{K-1} s_{k,j}w_j$ .

*Proof.* 1. For any  $v, w \in S_K^{(n)}$ , recalling notation (2.3) we have

$$\begin{aligned} (\mathcal{C}v, w)_{S_K^{(n)}} &= \sum_{j=1}^{K-1} [2c_0v_j + c_n(v_{j-1} + v_{j+1}) + \check{c} \cdot v_{j-1/2} + c \cdot v_{j+1/2}]w_j \\ &\quad + \sum_{j=1}^K (cv_{j-1} + \tilde{C}v_{j-1/2} + \check{c}v_{j+1}) \cdot w_{j-1/2}. \end{aligned} \quad (3.15)$$

2. According to formulas (3.15) and (3.3), for even  $e^{(l)}$ , we get

$$\begin{aligned} (\mathcal{C}s_0^{(l)}, w)_{S_K^{(n)}} &= \sum_{j=1}^{K-1} (-1)^j (c - \check{c}) \cdot e^{(l)}w_j + \sum_{j=1}^K (-1)^{j-1} \tilde{C}e^{(l)} \cdot (qw_{j-1} + \check{q}w_j) \\ &= \tilde{C}e^{(l)} \cdot (q - \check{q}) \sum_{j=1}^{K-1} (-1)^j w_j = 0 \end{aligned}$$

since  $(c - \check{c}) \cdot e^{(l)} = 0$  and  $\tilde{C}e^{(l)} \cdot (q - \check{q}) = 0$  for any  $c, q \in \mathbb{R}^{n-1}$  as well as  $w_0 = w_K = 0$ .

For odd  $e^{(l)}$ , we similarly get

$$(\mathcal{C}s_0^{(l)}, w)_{S_K^{(n)}} = \sum_{j=1}^{K-1} (c + \check{c}) \cdot e^{(l)}w_j + \sum_{j=1}^K \tilde{C}e^{(l)} \cdot (qw_{j-1} + \check{q}w_j) = \tilde{C}e^{(l)} \cdot (q + \check{q}) \sum_{j=1}^{K-1} w_j = 0$$

since  $(c + \check{c}) \cdot e^{(l)} = 0$  and  $\tilde{C}e^{(l)} \cdot (q + \check{q}) = 0$  for any  $c, q \in \mathbb{R}^{n-1}$  as well as  $w_0 = w_K = 0$ . Equality (3.13) is proved.

3. Formula (3.15) together with (3.4) and (2.5) imply that

$$\begin{aligned} (\mathcal{C}s_k^{(l)}, w)_{S_K^{(n)}} &= \sum_{j=1}^{K-1} [2(c_0 + c \cdot p_k^{(l)})s_{k,j} + (c_n + c \cdot \check{p}_k^{(l)})(s_{k,j-1} + s_{k,j+1})]w_j \\ &\quad + \sum_{j=1}^K [(\tilde{C}p_k^{(l)} + c)s_{k,j-1} + (\tilde{C}\check{p}_k^{(l)} + \check{c})s_{k,j}] \cdot w_{j-1/2} =: \sum' + \sum''. \end{aligned} \quad (3.16)$$

Owing to formula

$$s_{k,j-1} + s_{k,j+1} = 2\theta_k s_{k,j} \quad (3.17)$$

we first derive

$$\sum' = 2[c_0 + c \cdot p_k^{(l)} + \theta_k(c_n + c \cdot \check{p}_k^{(l)})](s_k, w)_{\omega_h}. \quad (3.18)$$

Second, using formulas (2.5) and (2.10), we get

$$\sum'' = \sum_{j=1}^K (\tilde{C}p_k^{(l)} + c) \cdot q(s_{k,j-1}w_{j-1} + s_{k,j}w_j) + (\tilde{C}p_k^{(l)} + c) \cdot \check{q}(s_{k,j-1}w_j + s_{k,j}w_{j-1}).$$

Owing to  $s_{k,0} = s_{k,K} = 0$ ,  $w_0 = w_K = 0$  as well as formulas (3.17), we further derive

$$\begin{aligned} \sum'' &= 2(\tilde{C}p_k^{(l)} + c) \cdot q(s_k, w)_{\omega_h} + (\tilde{C}p_k^{(l)} + c) \cdot \check{q} \sum_{j=1}^{K-1} (s_{k,j-1} + s_{k,j+1})w_j \\ &= 2[(\tilde{C}p_k^{(l)} + c) \cdot q + \theta_k(\tilde{C}p_k^{(l)} + c) \cdot \check{q}](s_k, w)_{\omega_h}. \end{aligned} \quad (3.19)$$

Adding (3.18) and (3.19), we prove (3.14).  $\square$

**Corollary 3.1.** *The orthogonality property (3.5) from Theorem 3.1, Item 4 is valid.*

*Proof.* It remains to consider the case  $k, \tilde{k} \in \overline{1, K-1}$  and  $k \neq \tilde{k}$ . Since then  $(s_k, s_{\tilde{k}})_{\omega_h} = 0$ , the result follows from (3.14).  $\square$

We call the calculation of  $w \in S_K^{(n)}$  by the coefficients  $w_{kl}$  of the expansion (3.6) as *the inverse  $F_n$ -transform* and the calculation of the coefficients  $w_{kl}$  by  $w \in S_K^{(n)}$  as *the direct  $F_n$ -transform*. We also consider related expansion of  $y \in S_K^{(n)}$

$$y = \sum_{l=1}^{n-1} \tilde{y}_{0l} \mathcal{C}s_0^{(l)} + \sum_{k=1}^{K-1} \sum_{l=1}^n \tilde{y}_{kl} \mathcal{C}s_k^{(l)} \quad (3.20)$$

and the calculation of the coefficients  $\tilde{y}_{kl}$  by  $y \in S_K^{(n)}$  that we call as *the direct  $FC_n$ -transform*.

Let us describe their fast FFT-based implementation.

**Theorem 3.2.** 1. *The inverse  $F_n$ -transform can be implemented according to the following formulas*

$$w_j = \sum_{k=1}^{K-1} \left( \sum_{l=1}^n w_{kl} \right) \sin \frac{\pi k j}{K}, \quad j = \overline{1, K-1}, \quad (3.21)$$

$$\begin{aligned} w_{j-1/2} &= (-P)^{j-1} \sum_{l=1}^{n-1} w_{0l} e^{(l)} \\ &+ 2 \sum_{k=1}^{K-1} d_{k,e} \cos \frac{\pi k}{2K} \sin \frac{\pi k(j-1/2)}{K} - 2 \sum_{k=1}^{K-1} d_{k,o} \sin \frac{\pi k}{2K} \cos \frac{\pi k(j-1/2)}{K}, \quad j = \overline{1, K}, \end{aligned} \quad (3.22)$$

where  $d_{k,e}$  and  $d_{k,o}$  are respectively even and odd components of the vector  $d_k := \sum_{l=1}^n w_{kl} p_k^{(l)}$ . Note that  $(-P)^{j-1}e = e$  for odd  $j$  and  $(-P)^{j-1}e = -\check{e}$  for even  $j$  for any  $e \in \mathbb{R}^{n-1}$ .

The collection  $\{w_j\}_{j=1}^{K-1}$  can be computed by the standard inverse FFT with respect to sines. The collection  $\{w_{j-1/2}\}_{j=1}^K$  can be computed by  $n-1$  modified inverse FFT related to the centers of elements in the amount of  $[n/2]$  with respect to sines and  $[(n-1)/2]$  with respect to cosines using extensions  $d_{K,e} := 0$  and  $d_{0,o} := 0$ , see algorithms DST-I, DST-III and DCT-III in [3].

2. *The direct  $FC_n$ -transform can be implemented basing on the standard formula*

$$\tilde{y}_{kl} = (y, s_k^{(l)})_{S_K^{(n)}} / \|s_k^{(l)}\|_{\mathcal{C}}^2. \quad (3.23)$$



Here, first, for  $k = 0$ ,  $l = \overline{1, n-1}$ , the following formulas hold

$$(y, s_0^{(l)})_{S_K^{(n)}} = \left( \sum_{j=1}^K (-P)^{j-1} y_{j-1/2} \right) \cdot e^{(l)}, \quad \|s_0^{(l)}\|_{\mathcal{C}}^2 = K. \quad (3.24)$$

Second, for  $k = \overline{1, K-1}$ ,  $l = \overline{1, n}$ , the following formulas hold

$$(y, s_k^{(l)})_{S_K^{(n)}} = \sum_{j=1}^{K-1} y_j \sin \frac{\pi k j}{K} + p_{k,e}^{(l)} \cdot \sum_{j=1}^{K-1} (y_{j-1/2} + y_{j+1/2})_e \sin \frac{\pi k j}{K} + p_{k,o}^{(l)} \cdot \sum_{j=1}^{K-1} (y_{j+1/2} - y_{j-1/2})_o \sin \frac{\pi k j}{K}, \quad (3.25)$$

$$\|s_k^{(l)}\|_{\mathcal{C}}^2 = K \left\{ c_0 + (\tilde{C} p_k^{(l)} + 2c) \cdot p_k^{(l)} + \theta_k [c_n + (\tilde{C} p_k^{(l)} + 2c) \cdot \check{p}_k^{(l)}] \right\}. \quad (3.26)$$

Notice that the sums in formula (3.25) are independent on  $l$ . The collection of all these coefficients can be computed using  $n$  standard direct FFTs with respect to sines.

3. Similarly to Item 2, the direct  $F_n$ -transform can be implemented basing on the standard formula

$$w_{kl} = (\mathcal{C}w, s_k^{(l)})_{S_K^{(n)}} / \|s_k^{(l)}\|_{\mathcal{C}}^2. \quad (3.27)$$

Here, for  $k = 0$ ,  $l = \overline{1, n-1}$ , the following formula holds

$$(\mathcal{C}w, s_0^{(l)})_{S_K^{(n)}} = \left( \tilde{C} \sum_{j=1}^K (-P)^{j-1} w_{j-1/2} \right) \cdot e^{(l)}. \quad (3.28)$$

For  $k = \overline{1, K-1}$ ,  $l = \overline{1, n}$ , formula (3.25) with  $y := \mathcal{C}w$  is applicable. Alternatively, the following formula holds as well

$$(\mathcal{C}w, s_k^{(l)})_{S_K^{(n)}} = 2[c_0 + c \cdot p_k^{(l)} + \theta_k(c_n + c \cdot \check{p}_k^{(l)})] \sum_{j=1}^{K-1} w_j \sin \frac{\pi k j}{K} + q_{k,e}^{(l)} \cdot \sum_{j=1}^{K-1} (w_{j-1/2} + w_{j+1/2})_e \sin \frac{\pi k j}{K} + q_{k,o}^{(l)} \cdot \sum_{j=1}^{K-1} (w_{j+1/2} - w_{j-1/2})_o \sin \frac{\pi k j}{K}, \quad (3.29)$$

where  $q_{k,e}^{(l)}$  and  $q_{k,o}^{(l)}$  are respectively even and odd components of the vector  $q_k^{(l)} := \tilde{C} p_k^{(l)} + c$ . Once again all these coefficients can be computed using  $n$  standard direct FFTs with respect to sines.

*Proof.* 1. Let the coefficients  $w_{kl}$  of expansion (3.6) be known. According to the first formulas (3.3) and (3.4), the values of  $w$  for integer indices in (3.6) are reduced to (3.21).

To compute  $w$  for half-integer indices, we transform the second sum in (3.6). Owing to decomposition (2.4) we rewrite the second formula (3.4) in the form

$$s_{k,j-1/2}^{(l)} = p_{k,e}^{(l)} \left( \sin \frac{\pi k(j-1)}{K} + \sin \frac{\pi k j}{K} \right) - p_{k,o}^{(l)} \left( \sin \frac{\pi k j}{K} - \sin \frac{\pi k(j-1)}{K} \right) \\ = 2 \cos \frac{\pi k}{2K} p_{k,e}^{(l)} \sin \frac{\pi k(j-1/2)}{K} - 2 \sin \frac{\pi k}{2K} p_{k,o}^{(l)} \cos \frac{\pi k(j-1/2)}{K}, \quad j = \overline{1, K}.$$

Then using also the second formula (3.3), we obtain formula (3.22).

2. Now we consider the computation of the coefficients in expansion (3.20) for given  $y \in S_K$ . Owing to the orthogonality property (3.5), they first can be expressed in the form (3.23) for  $k = 0$ ,  $l = \overline{1, n-1}$  and  $k = \overline{1, K-1}$ ,  $l = \overline{1, n}$ .

Formulas (3.15) and (3.3) imply that

$$\|s_0^{(l)}\|_{\mathcal{C}}^2 = K\tilde{C}e^{(l)} \cdot e^{(l)} = K, \quad l = \overline{1, n-1}.$$

Lemma 3.1 immediately implies formula (3.26) since  $(s_k, s_k)_{\omega_h} = K/2$ .

By virtue of formulas (3.3) for the numerator of formula (3.23) for  $k = 0$  we can write

$$(y, s_0^{(l)})_{S_K^{(n)}} = \sum_{j=1}^K y_{j-1/2} \cdot (-P)^{j-1} e^{(l)} = \left( \sum_{j=1}^K (-P)^{j-1} y_{j-1/2} \right) \cdot e^{(l)}.$$

By virtue of formulas (3.4) for the same numerator for  $k = \overline{1, K-1}$  we get

$$(y, s_k^{(l)})_{S_K^{(n)}} = \sum_{j=1}^{K-1} y_j s_{k,j} + \sum_{j=1}^K y_{j-1/2} s_{k,j-1} \cdot p_k^{(l)} + \sum_{j=1}^K y_{j-1/2} s_{k,j} \cdot \check{p}_k^{(l)}. \quad (3.30)$$

Therefore shifting by 1 the index in the second of these sums, and applying the identity  $a_1 b_1 + a_2 b_2 = 0.5(a_1 + a_2)(b_1 + b_2) + 0.5(a_1 - a_2)(b_1 - b_2)$  and recalling decomposition (2.4), we derive

$$\begin{aligned} & (y, s_k^{(l)})_{S_K^{(n)}} \\ &= \sum_{j=1}^{K-1} y_j s_{k,j} + p_{k,e}^{(l)} \cdot \sum_{j=1}^{K-1} (y_{j-1/2} + y_{j+1/2}) s_{k,j} + p_{k,o}^{(l)} \cdot \sum_{j=1}^{K-1} (y_{j+1/2} - y_{j-1/2}) s_{k,j}. \end{aligned}$$

Since also

$$p_e \cdot q = p_e \cdot q_e, \quad p_o \cdot q = p_o \cdot q_o \quad \text{for any } q, p \in \mathbb{R}^{n-1},$$

we obtain formula (3.25).

3. Owing to the orthogonality property (3.5), formula (3.27) is valid.

By virtue of formulas (3.3), (3.15) as well as  $\tilde{C}^T = \tilde{C}$  and  $P = P^T$  for its numerator for  $k = 0$  we can write

$$(\mathcal{C}w, s_0^{(l)})_{S_K^{(n)}} = (\mathcal{C}s_0^{(l)}, w)_{S_K^{(n)}} = \sum_{j=1}^K \tilde{C}(-P)^{j-1} e^{(l)} \cdot w_{j-1/2} = \left( \tilde{C} \sum_{j=1}^K (-P)^{j-1} w_{j-1/2} \right) \cdot e^{(l)}.$$

By virtue of formulas (3.16), (3.18) for the same numerator for  $k = \overline{1, K-1}$  we get

$$\begin{aligned} & (\mathcal{C}w, s_k^{(l)})_{S_K^{(n)}} = (\mathcal{C}s_k^{(l)}, w)_{S_K^{(n)}} \\ &= 2[c_0 + c \cdot p_k^{(l)} + \theta_k(c_n + c \cdot \check{p}_k^{(l)})](s_k, w)_{\omega_h} + \sum_{j=1}^K (q_k^{(l)} s_{k,j-1} + \check{q}_k^{(l)} s_{k,j}) \cdot w_{j-1/2}, \end{aligned}$$

where  $q_k^{(l)} = \tilde{C}p_k^{(l)} + c$ . Transforming the last sum in the same manner as above the second and third terms in (3.30), we obtain (3.29).  $\square$

## 4 Applications to the generalized Poisson equation

To begin with, we turn to the simple 1D ODE boundary value problem

$$-u''(x) + \alpha u(x) = f(x) \quad \text{on } [0, X], \quad u(0) = u(X) = 0, \quad (4.1)$$

where for simplicity  $\alpha = \text{const} > -(\pi/X)^2$ . Its FEM discretization has the operator form

$$4h^{-2}\mathcal{A}v + \alpha\mathcal{C}v = f^h, \quad v \in S_K^{(n)}, \quad (4.2)$$

where  $f^h \in S_K^{(n)}$  is the FEM average of  $f$ . Its solution can be written in the form

$$v = \sum_{k=0}^K \sum_{l=1}^{n-\delta_{k0}} \frac{\tilde{f}_{kl}^h}{4h^{-2}\lambda_k^{(l)} + \alpha} s_k^{(l)}, \quad (4.3)$$

of the expansion like (3.6), where  $\tilde{f}_{kl}^h$  are the coefficients of the expansion like (3.20) for the vector  $f^h$ ; recall that  $\delta_{k0}$  is the Kronecker delta.

Next we consider in detail solving of the  $N$ -dimensional ( $N \geq 2$ ) boundary value problem

$$-\Delta u + \alpha u = f \quad \text{in } \Omega = (0, X_1) \times \dots \times (0, X_N), \quad u|_{\partial\Omega} = 0, \quad (4.4)$$

where  $\Delta$  is the Laplace operator and  $\alpha = \text{const} > -\pi^2(X_1^{-2} + \dots + X_N^{-2})$  (for simplicity, in order to treat the positive definite operator that actually is not so necessary).

We define the space  $H_{h_1}^{(n_1)}[0, X_1] \otimes \dots \otimes H_{h_N}^{(n_N)}[0, X_N]$  of the piecewise-polynomial in  $\bar{\Omega}$  functions, where  $h_i = X_i/K_i$  and  $n_i \geq 2, i = \overline{1, N}$ . Let  $\mathbf{K} = (K_1, \dots, K_N)$  and  $\mathbf{n} = (n_1, \dots, n_N)$ .

We also define the space  $S_{\mathbf{K}}^{(\mathbf{n})} = S_{K_1}^{(n_1)} \otimes \dots \otimes S_{K_N}^{(n_N)}$  of vector functions. For example, for  $N = 2$ , these functions are numbers for the indices  $(j_1, j_2)$ ,  $j_1 = \overline{0, K_1}$ ,  $j_2 = \overline{0, K_2}$ , and vectors from  $\mathbb{R}^{n_1-1}$ ,  $\mathbb{R}^{n_2-1}$  and  $\mathbb{R}^{(n_1-1) \times (n_2-1)}$  respectively for the indices

$$(j_1 - 1/2, j_2), j_1 = \overline{1, K_1}, j_2 = \overline{0, K_2}; \quad (j_1, j_2 - 1/2), j_1 = \overline{0, K_1}, j_2 = \overline{1, K_2} \quad \text{and} \\ (j_1 - 1/2, j_2 - 1/2), j_1 = \overline{1, K_1}, j_2 = \overline{1, K_2},$$

as well as zero vectors for  $j_1 = 0, K_1$  and  $j_2 = 0, K_2$ . Similarly to the 1D case, there is the natural isomorphism between functions in  $H_{h_1}^{(n_1)}[0, X_1] \otimes \dots \otimes H_{h_N}^{(n_N)}[0, X_N]$  and vectors in  $S_{\mathbf{K}}^{(\mathbf{n})}$ .

The FEM discretization of problem (4.4) can be written in the following operator form

$$(4h_1^{-2}\mathcal{A}_1\mathcal{C}_2 \dots \mathcal{C}_N + \dots + 4h_N^{-2}\mathcal{A}_N\mathcal{C}_1 \dots \mathcal{C}_{m-1})v + \alpha\mathcal{C}_1 \dots \mathcal{C}_N v = f^h, \quad v \in S_{\mathbf{K}}^{(\mathbf{n})}, \quad (4.5)$$

where  $\mathcal{A}_i$  and  $\mathcal{C}_i$  are versions of the above defined operators  $\mathcal{A}$  and  $\mathcal{C}$  acting in variable  $x_i$  (depending on  $K_i$  and  $n_i$ ),  $i = \overline{1, N}$ , and  $f^h \in S_{\mathbf{K}}^{(\mathbf{n})}$  is the FEM average of  $f$ . Recall that the case of the non-homogeneous Dirichlet boundary condition  $u(x) = b(x)$  on  $\partial\Omega$  in (4.4) could be easily covered by reducing to (4.5) with the modified  $f^h$  depending on an approximation  $b^h$  of  $b$ .

To compute its solution, the  $F_n$ -transforms from Theorem 3.2 can be applied twofold.

(a) We consider the multiple expansion of  $f^h \in S_{\mathbf{K}}^{(\mathbf{n})}$  like (3.20)

$$f^h = \sum_{i=1}^N \sum_{k_i=0}^{K_i-1} \sum_{l_i=1}^{n_i-\delta_{k_i0}} \tilde{f}_{k_1 l_1, \dots, k_N l_N}^h \mathcal{C}_1 s_{1, k_1}^{(l_1)} \dots \mathcal{C}_N s_{N, k_N}^{(l_N)}. \quad (4.6)$$

Then the expansion of the solution has the following form

$$v = \sum_{i=1}^N \sum_{k_i=0}^{K_i-1} \sum_{l_i=1}^{n_i-\delta_{k_i0}} \frac{\tilde{f}_{k_1 l_1, \dots, k_N l_N}^h}{4h_1^{-2}\lambda_{1, k_1}^{(l_1)} + \dots + 4h_N^{-2}\lambda_{N, k_N}^{(l_N)} + \alpha} s_{1, k_1}^{(l_1)} \dots s_{N, k_N}^{(l_N)}. \quad (4.7)$$

Here  $\{\lambda_{i, k_i}^{(l_i)}, s_{i, k_i}^{(l_i)}\}$  are versions of the above defined eigenpairs  $\{\lambda_k^{(l)}, s_k^{(l)}\}$  with respect to  $x_i$ .

Algorithm (a) comprises two rather standard steps:

- (1) finding the coefficients of expansion (4.6) for  $f^h$  (by the direct  $FC_n$ -transforms in  $x_1, \dots, x_N$ );
  - (2) finding  $v$  by the coefficients of its expansion (4.7) (by the inverse  $F_n$ -transforms in  $x_1, \dots, x_N$ ).
- (b) We consider the expansion of  $f^h$  like (3.20) in  $x_2, \dots, x_N$ , i.e.

$$f^h = \sum_{i=2}^N \sum_{k_i=0}^{K_i-1} \sum_{l_i=1}^{n_i-\delta_{k_i0}} \tilde{f}_{k_2 l_2, \dots, k_N l_N}^h \mathcal{C}_2 s_{2, k_2}^{(l_2)} \dots \mathcal{C}_N s_{N, k_N}^{(l_N)}, \quad (4.8)$$

now with the coefficients  $\tilde{f}_{k_2 l_2, \dots, k_N l_N}^h \in S_{K_1}^{(n_1)}$ . Then the coefficients  $v_{kl} \in S_{K_1}^{(n_1)}$  in the similar expansion of the solution  $v \in S_{\mathbf{K}}^{(\mathbf{n})}$

$$v = \sum_{i=2}^N \sum_{k_i=0}^{K_i-1} \sum_{l_i=1}^{n_i-\delta_{k_i0}} v_{k_2 l_2, \dots, k_N l_N} s_{2, k_2}^{(l_2)} \dots s_{m, k_N}^{(l_N)}, \quad (4.9)$$

serve as the solutions to 1D problems in  $x_1$

$$[4h_1^{-2}\mathcal{A}_1 + (4h_2^{-2}\lambda_{k_2}^{(l_2)} + \dots + 4h_N^{-2}\lambda_{k_N}^{(l_N)} + \alpha)\mathcal{C}_1]v_{k_2 l_2, \dots, k_N l_N} = \tilde{f}_{k_2 l_2, \dots, k_N l_N}^h. \quad (4.10)$$

Their matrices are symmetric and positive definite.

Algorithm (b) comprises three rather standard steps:

- (1) finding the coefficients of the expansion (4.8) for  $f^h$  (by the direct  $FC_n$ -transforms in  $x_2, \dots, x_N$ );
- (2) solving the collection of the independent 1D problems (4.10) for the coefficients of the expansion of  $v$ ;
- (3) finding  $v$  by the coefficients of its expansion (4.9) (by the inverse  $F_n$ -transforms in  $x_2, \dots, x_N$ ).

Implementing algorithms (a) and (b) costs respectively  $O(K_1 \dots K_N \log_2(K_1 \dots K_N))$  and  $O(K_1 \dots K_N \log_2(K_2 \dots K_N))$  arithmetic operations.

Importantly, they can be applied to solve various time-dependent PDEs such as the heat, wave or Schrödinger's equations since for their implicit time discretizations one usually gets problems like (4.5) at the upper time level.

Moreover, algorithm (b) is directly extended to the case of more general equations than in (4.4) with the coefficients depending on  $x_1$  (that is essential, in particular, in the polar and cylindrical coordinates), various boundary conditions for  $x_1 = 0, X_1$  and the nonuniform mesh in  $x_1$  [11]. It can also be applied for reducing 3D problems in a cylindrical domain to a collection of independent 2D problems in the cylinder base.

## 5 Numerical experiments

1. We first check that the eigenvalues of each of problems (2.8) and (2.9) are well separated. We define their spectral gaps as

$$\min_{1 \leq l \leq n} (\lambda_{0(l+1)} - \lambda_{0(l)}) = \frac{\pi^2}{4} + \delta_n, \quad \min_{1 \leq l \leq n-2} (\lambda_0^{(l+1)} - \lambda_0^{(l)}) = \frac{3\pi^2}{4} + \tilde{\delta}_n,$$

where  $S_n =: \{\lambda_{0(l)}\}_{l=1}^{n+1}$  and present  $\delta_n$  and  $\tilde{\delta}_n$  in Fig. 1 (left). The terms  $\pi^2/4$  and  $3\pi^2/4$  are the spectral gaps (in fact, the gaps between two minimal eigenvalues) of the corresponding ODE problems, see [13]. We observe that both  $\delta_n$  and  $\tilde{\delta}_n$  are decreasing and rapidly tend to 0 as  $n$  increases. We also checked that  $S_n \cap \tilde{S}_n = \emptyset$  for all  $2 \leq n \leq 21$ .

Also we give the spectral condition numbers  $\text{cond } \tilde{A}$  and  $\text{cond } \tilde{C}$  in Fig. 1 (right) and remark their rapid growth as  $n$  increases (unfortunately).

Notice that all our computations are accomplished on an ordinary notebook ASUS-U36S with Intel Core i3-2350M CPU 2.3 GHz, 8 Gb, Win 10 x64. The codes in Matlab R2016a were developed to implement the algorithms, and we emphasize that several basic and advanced [1] code vectorisation techniques were applied to speed up them notably.

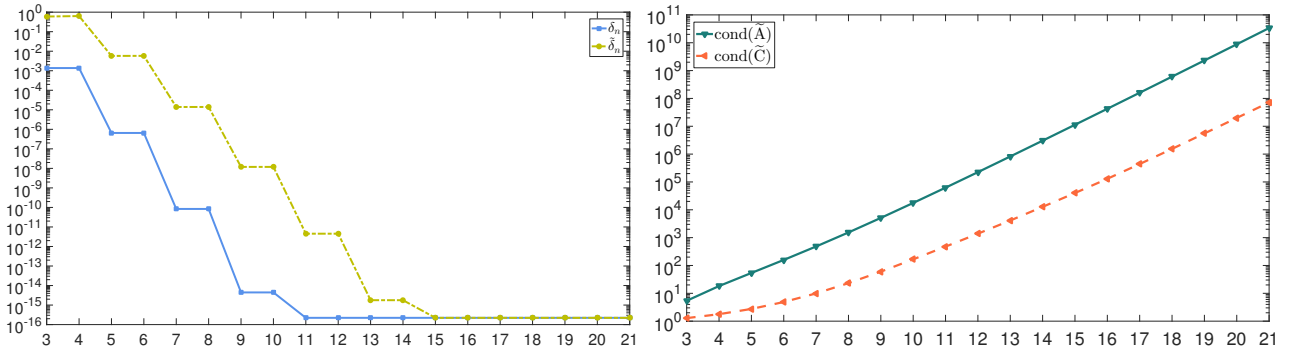


Figure 1: The numbers  $\delta_n$  and  $\tilde{\delta}_n$  related to the minimal spectral gaps of eigenvalue problems (2.8) and (2.9) (left) as well as  $\text{cond } \tilde{A}$  and  $\text{cond } \tilde{C}$  (right) for problem (2.9) in dependence on  $n$

In the case of the 1D ODE problem (4.2) for  $\alpha = 1$ , we provide the condition numbers of the global FEM matrix  $4h^{-2}\mathcal{A} + \alpha\mathcal{C}$  for  $\alpha = 1$  and its local version in dependence on  $K = 2, 4, \dots, 1024$  for  $n = \overline{1, 9}$  in Fig. 2 for a further reference. Notice their rather rapid growth as  $K$  increases.

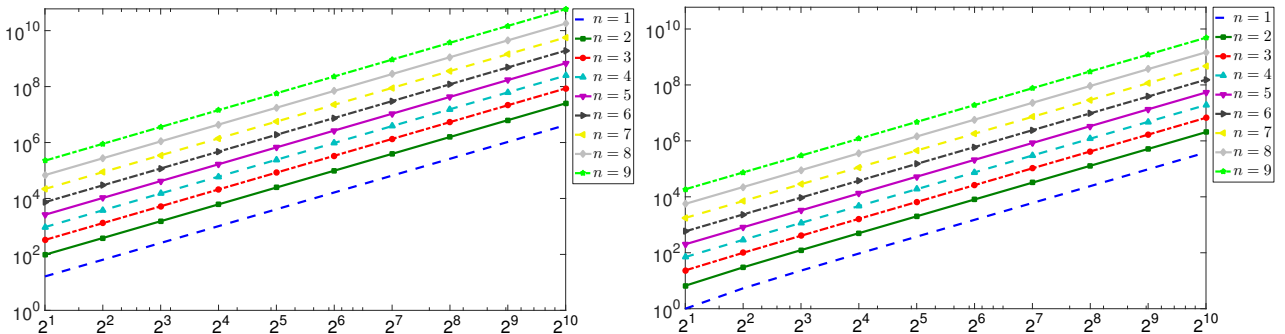


Figure 2: The condition numbers of the local (left) and global (right) matrices in 1D problem (4.2) for  $\alpha = 1$  in dependence on  $K = 2, 4, \dots, 1024$  for  $n = \overline{1, 9}$

Next we analyze the execution time for  $F_n$ -transforms in dependence on the choice of  $K$  as it grows up to very high values  $2^{20}$ . We consider three choices of  $K$ : powers of two ( $K = 2^p$ ),

primes or other composite numbers (not powers of two). Primes and composite values of  $K$  are randomly chosen as different ten numbers between each two consecutive powers of two, and the comparison is accomplished in the spirit of [8]. We use the external function `comp_dst` for DST-I and DST-III and `comp_dct` for DCT-III from Large Time-Frequency Analysis Toolbox [15] based on FFTW library [9], which is also used in the **Matlab** function `fft` for FFT.

The execution times for the DST-I and DST-III as well as our transforms in the case  $n = 5$  (for definiteness) are given respectively in Fig. 3 and Fig. 4. We do not take into account the execution time for the pairs  $\{\lambda_k^{(l)}, p_k^{(l)}\}$  since it becomes negligible for multiple using of the transforms for fixed  $K$  as required below. For our transforms the execution times are respectively larger (than for the DST-I and DST-III) due to multiple using of FFTs and some additional computations. The inverse  $F_n$ -transform takes less time than the direct one. Moreover, the best results are mainly for  $K = 2^p$  though this is not the case for DST-I (fortunately, we apply it in the optimal case  $K = 2^p - 1$ ). For the two other choices of  $K$  the execution times are worse but close, and the difference between all of them is less than in the case of both DST-I and DST-III. These results are attractive. Notice that the data in Fig. 4 can be approximated by linear functions for  $2^5 \leq K \leq 2^{10}$  and  $2^{10} \leq K \leq 2^{20}$  but with rather flat slope in the former case and a visibly higher slope in the latter one.

**Remark 5.1.** *The last phenomenon is explained by the advanced architecture of modern processors involving cache memory, streaming SIMD extensions and advanced vector extensions of the instruction set, etc. Also the above mentioned high-quality implementations of FFTs are applied.*

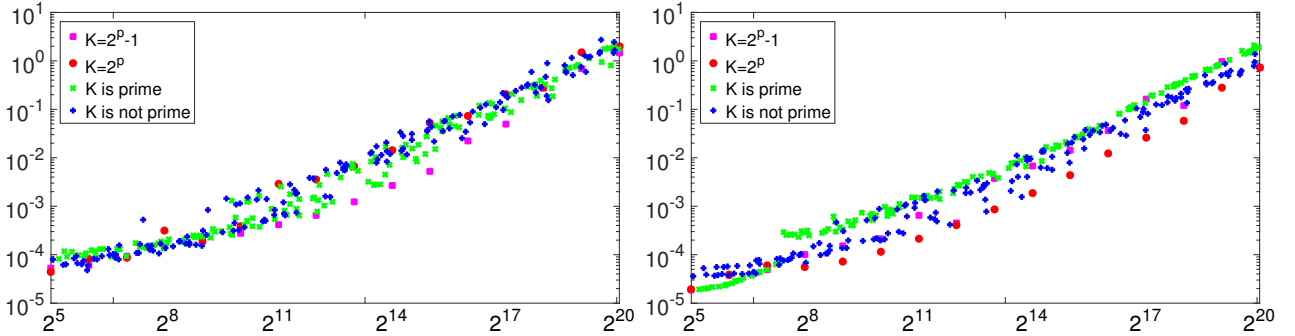


Figure 3: The execution time (in seconds) for DST-I (left) and DST-III (right) for the several choices of  $K$  in between  $2^5$  and  $2^{20}$

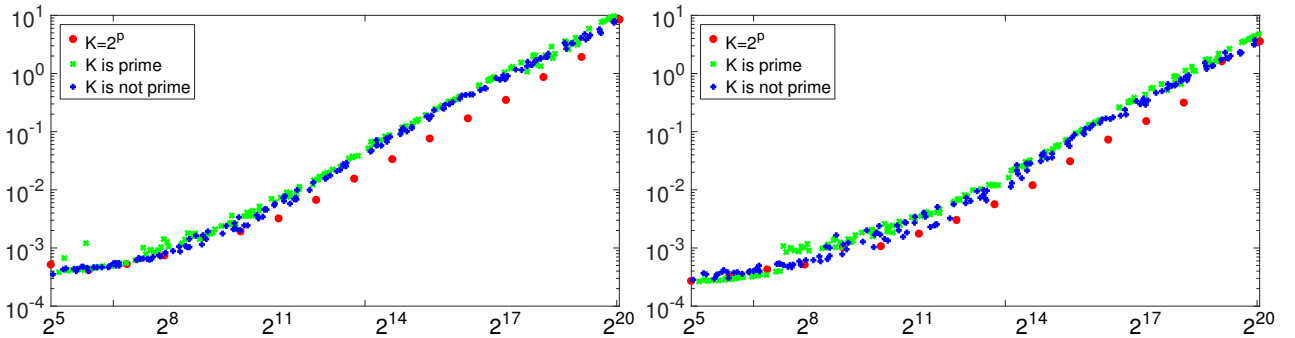


Figure 4: The execution time (in seconds) for the direct (left) and inverse (right)  $F_n$ -transforms,  $n = 5$ , for the several choices of  $K$  in between  $2^5$  and  $2^{20}$

2. Our main computational results concern solving problem (4.4) for both  $N = 2$  and  $3$ ,  $\alpha = 1$  and  $X_i = 1$ . We take  $K_i = K$  and  $n_i = n$  for simplicity. We apply the multiple Gauss quadrature formulas with  $n + 1$  nodes in  $x_i$  to compute  $f^h$ . The eigenpairs of the 1D problems are computed with the quadruple precision (using Mathematica) to improve the stability with respect to round-off errors. We notice that system (4.5) contains  $(Kn - 1)^N$  unknowns. For comparison, we include the simplest known case  $n = 1$  implemented in the code in the unified manner.

We first consider the 2D case ( $N = 2$ ) and take the exact solution  $u(x) := \sin(2\pi x_1) \sin(3\pi x_2) \cosh(\sqrt{2}x_1 - x_2)$ . We compute the FEM solution for different values of  $n$  and  $K$  in order to study the (absolute) error behaviour, see Fig. 5 and Table 1 (where values of  $R_C$  less than 3 are omitted). To reduce the possible round-off errors, hereafter we compute the pairs  $\{\lambda_k^{(l)}, p_k^{(l)}\}$  with the quadruple precision. For algorithm (a), the error behaviour in the uniform mesh norm is standard: the rate of its decreasing  $R_C$  is mainly proportional to  $(n + 1)^2$  except for  $n = 2$  when it is faster and similar to  $n = 3$  (the exception has previously been noted in [5]). Curiously, for  $n = 9$  and the minimal  $K = 2$  the error is already less than for  $n = 1$  and the maximal  $K = 1024$ . Of course, the error cannot be better than the level of round-off errors that is achieved the faster, the higher  $5 \leq n \leq 9$ . We observe that impact of the round-off errors is almost absent. For algorithm (b), the situation is similar only up to the error level  $\sim 10^{-11}$ . Once this value is reached for fixed  $3 \leq n \leq 9$ , we further see the error growth as  $K$  increases that means the perceptible impact of the round-off errors. This is due to the respective growth of condition numbers for matrices in system (4.10) as  $K$  or  $n$  increases, see above Fig. 2. Consequently algorithm (a) is preferable than (b) provided that very high precision is required. Notice that if we use the double precision only, then the level of the best error becomes at  $\sim 10^{-12} - 10^{-13}$  and the results remain stable for algorithm (a) but they remain practically unchanged for algorithm (b). Thus only the double precision computations are possible provided that the mentioned accuracy is sufficient (that is the case in a lot of applications).

We also analyze the execution time for both algorithms (using multiple program runs and their median execution time), see Fig. 6 and Tables 2 and 3 for the same  $K$  and  $n$ . Clearly this time is independent of the above specific choice of  $u$ . We do not take into account the execution time for the pairs  $\{\lambda_k^{(l)}, p_k^{(l)}\}$  considering the case where they are computed in advance and stored (recall that they are independent of the data of PDE problem (4.4)). We call attention to the weakly superlinear behaviour of time in  $K$  and its mild monotone growth in  $n$ . Notice that all the ratios of the consecutive execution times in the both tables are even less than the lower bound 4 for the theoretical ratios (the ratios less than one are omitted); see Remark 5.1 in this respect. In contrast to theoretical expectations, almost all ratios for algorithm (a) are less than for (b). The ratios grows as  $K$  and  $n$  increase, and for algorithm (b) the highest value is already close to 4. For the maximal  $K = 1024$  and  $n = 9$ , system (4.5) contains almost  $85 \cdot 10^6$  unknowns but only less than 2 min is required to solve it that is the excellent result.

Finally we consider the most interesting 3D case ( $N = 3$ ) and take the exact solution  $u(x) := \sin(2\pi x_1) \sin(3\pi x_2) \sin(4\pi x_3) \cosh(\sqrt{2}x_1 - x_2 + x_3/\sqrt{3})$ . Once again we compute the FEM solution for different values of  $K = 2, 4, \dots, 64$  and  $n = \overline{1, 9}$  and study the error behaviour, see Fig. 7 and Table 4 (where values of  $R_C$  less than 3 are omitted). Conclusions are generally the same as in the 2D case. Notice that now the worse stability properties of algorithm (b) are visible only for  $n = 8, 9$  since much less maximal value of  $K$  is taken.

The execution times in the 3D case are presented in Fig. 8 and Tables 5 and 6. Once more conclusions are similar to the 2D case. All the ratios in the both tables are notably less than the lower bound 8 for the theoretical ratios. Importantly, for the maximal  $K = 64$  and  $n = 9$ , system (4.5) contains more than  $190 \cdot 10^6$  unknowns but only less than 15 min is required to

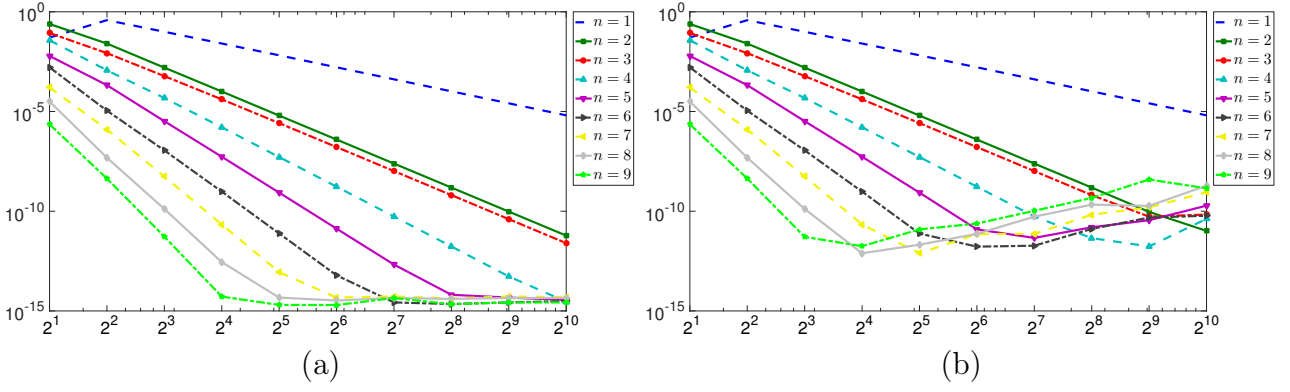


Figure 5: The 2D case. The errors in the mesh uniform norm for algorithms (a) and (b) in dependence on  $K = 2, 4, \dots, 1024$  for  $n = \overline{1, 9}$

$K$	$n = 1$	$R_C$	$n = 2$	$R_C$	$n = 3$	$R_C$	$n = 4$	$R_C$	$n = 5$	$R_C$
2	$5.1 \cdot 10^{-2}$	—	$2.4 \cdot 10^{-1}$	—	$8.7 \cdot 10^{-2}$	—	$3.7 \cdot 10^{-2}$	—	$6.1 \cdot 10^{-3}$	—
4	$3.8 \cdot 10^{-1}$	—	$2.5 \cdot 10^{-2}$	9.6	$8.4 \cdot 10^{-3}$	10.3	$1.2 \cdot 10^{-3}$	32.2	$2.1 \cdot 10^{-4}$	29.3
8	$1.0 \cdot 10^{-1}$	3.8	$1.6 \cdot 10^{-3}$	16.1	$5.9 \cdot 10^{-4}$	14.2	$4.7 \cdot 10^{-5}$	24.5	$3.3 \cdot 10^{-6}$	63.8
16	$2.6 \cdot 10^{-2}$	3.9	$1.0 \cdot 10^{-4}$	15.7	$4.1 \cdot 10^{-5}$	14.6	$1.6 \cdot 10^{-6}$	29.3	$5.4 \cdot 10^{-8}$	60.8
32	$6.6 \cdot 10^{-3}$	3.9	$6.2 \cdot 10^{-6}$	16.1	$2.6 \cdot 10^{-6}$	15.6	$5.2 \cdot 10^{-8}$	31.1	$8.5 \cdot 10^{-10}$	63.2
64	$1.6 \cdot 10^{-3}$	4.0	$3.9 \cdot 10^{-7}$	15.9	$1.6 \cdot 10^{-7}$	15.9	$1.7 \cdot 10^{-9}$	30.6	$1.3 \cdot 10^{-11}$	64.0
128	$4.1 \cdot 10^{-4}$	4.0	$2.4 \cdot 10^{-8}$	16.0	$1.0 \cdot 10^{-8}$	16.0	$5.4 \cdot 10^{-11}$	31.4	$2.1 \cdot 10^{-13}$	63.1
256	$1.0 \cdot 10^{-4}$	4.0	$1.5 \cdot 10^{-9}$	16.0	$6.4 \cdot 10^{-10}$	16.0	$1.7 \cdot 10^{-12}$	31.7	$6.4 \cdot 10^{-15}$	32.8
512	$2.6 \cdot 10^{-5}$	4.0	$9.6 \cdot 10^{-11}$	16.0	$4.0 \cdot 10^{-11}$	16.0	$5.4 \cdot 10^{-14}$	31.7	$4.7 \cdot 10^{-15}$	—
1024	$6.4 \cdot 10^{-6}$	4.0	$6.0 \cdot 10^{-12}$	16.0	$2.5 \cdot 10^{-12}$	16.0	$2.9 \cdot 10^{-15}$	18.6	$3.6 \cdot 10^{-15}$	—

$K$	$n = 6$	$R_C$	$n = 7$	$R_C$	$n = 8$	$R_C$	$n = 9$	$R_C$
2	$1.6 \cdot 10^{-3}$	—	$1.6 \cdot 10^{-4}$	—	$3.2 \cdot 10^{-5}$	—	$2.3 \cdot 10^{-6}$	—
4	$1.1 \cdot 10^{-5}$	148.2	$1.3 \cdot 10^{-6}$	129.3	$4.8 \cdot 10^{-8}$	657.8	$4.3 \cdot 10^{-9}$	521.5
8	$1.1 \cdot 10^{-7}$	98.0	$5.5 \cdot 10^{-9}$	229.1	$1.3 \cdot 10^{-10}$	373.8	$5.3 \cdot 10^{-12}$	817.1
16	$9.6 \cdot 10^{-10}$	117.1	$2.2 \cdot 10^{-11}$	254.6	$2.8 \cdot 10^{-13}$	459.0	$5.2 \cdot 10^{-15}$	1019.3
32	$7.6 \cdot 10^{-12}$	125.6	$8.8 \cdot 10^{-14}$	245.2	$4.7 \cdot 10^{-15}$	60.4	$2.0 \cdot 10^{-15}$	—
64	$6.1 \cdot 10^{-14}$	125.5	$4.7 \cdot 10^{-15}$	18.8	$3.3 \cdot 10^{-15}$	—	$2.0 \cdot 10^{-15}$	—
128	$2.7 \cdot 10^{-15}$	22.8	$5.3 \cdot 10^{-15}$	—	$4.4 \cdot 10^{-15}$	—	$4.4 \cdot 10^{-15}$	—
256	$2.2 \cdot 10^{-15}$	—	$4.2 \cdot 10^{-15}$	—	$4.0 \cdot 10^{-15}$	—	$2.2 \cdot 10^{-15}$	—
512	$2.7 \cdot 10^{-15}$	—	$5.3 \cdot 10^{-15}$	—	$4.4 \cdot 10^{-15}$	—	$2.7 \cdot 10^{-15}$	—
1024	$3.1 \cdot 10^{-15}$	—	$4.9 \cdot 10^{-15}$	—	$4.4 \cdot 10^{-15}$	—	$2.7 \cdot 10^{-15}$	—

Table 1: The 2D case. The errors in the mesh uniform norm and their ratios  $R_C$  in dependence on  $K = 2, 4, \dots, 1024$  and  $n = \overline{1, 9}$  for algorithm (a)



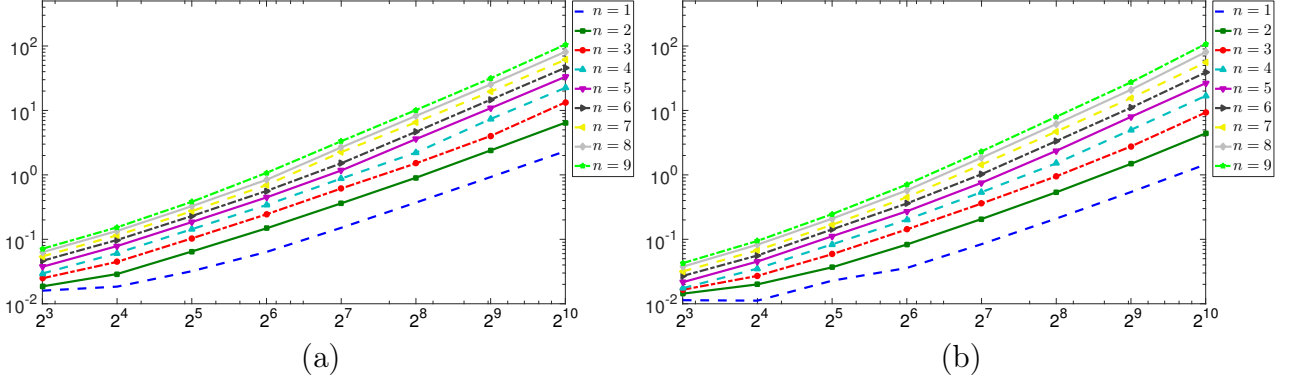


Figure 6: The 2D case. The execution time (in seconds) for algorithms (a) and (b) in dependence on  $K = 8, 16, \dots, 1024$  for  $n = \overline{1, 9}$

$K$	$n = 1$	$n = 2$	$n = 3$	$n = 4$	$n = 5$	$n = 6$	$n = 7$	$n = 8$	$n = 9$
8	—	—	—	1.46	1.6	1.97	2	2.02	2.01
16	1.15	1.55	1.8	2.07	2.1	2.1	2.13	2.14	2.15
32	1.74	2.24	2.31	2.35	2.35	2.35	2.37	2.42	2.49
64	1.99	2.29	2.37	2.38	2.41	2.45	2.55	2.57	2.81
128	2.37	2.44	2.51	2.57	2.62	2.69	3.24	3.19	3.1
256	2.46	2.49	2.46	2.53	3.07	3.08	2.9	3.08	3.05
512	2.49	2.66	2.66	3.33	3.02	3.15	2.99	3.08	3.11
1024	2.52	2.69	3.33	3.04	3.07	3.13	3.14	3.22	3.34

Table 2: The 2D case. The ratios of the execution times for algorithm (a) in dependence on  $K = 2, 4, \dots, 1024$  and  $n = \overline{1, 9}$

$K$	$n = 1$	$n = 2$	$n = 3$	$n = 4$	$n = 5$	$n = 6$	$n = 7$	$n = 8$	$n = 9$
8	—	—	—	1.17	1.28	1.97	1.99	1.99	2.01
16	—	1.39	1.64	2.02	2.09	2.08	2.13	2.2	2.22
32	2.04	1.84	2.17	2.38	2.47	2.54	2.49	2.53	2.61
64	1.57	2.26	2.42	2.42	2.44	2.52	2.67	2.74	2.88
128	2.34	2.49	2.52	2.67	2.77	2.85	3.21	3.21	3.27
256	2.5	2.6	2.64	2.8	3.13	3.27	3.25	3.34	3.44
512	2.58	2.76	2.91	3.31	3.37	3.29	3.31	3.4	3.43
1024	2.68	2.98	3.36	3.36	3.33	3.53	3.58	3.84	3.94

Table 3: The 2D case. The ratios of the execution times for algorithm (b) in dependence on  $K = 2, 4, \dots, 1024$  and  $n = \overline{1, 9}$

$K$	$n = 1$	$R_C$	$n = 2$	$R_C$	$n = 3$	$R_C$	$n = 4$	$R_C$	$n = 5$	$R_C$
2	$1.3 \cdot 10^{-2}$	—	$2.6 \cdot 10^{-2}$	—	$2.8 \cdot 10^{-1}$	—	$2.1 \cdot 10^{-1}$	—	$3.1 \cdot 10^{-2}$	—
4	$3.1 \cdot 10^{-2}$	—	$6.9 \cdot 10^{-2}$	—	$3.7 \cdot 10^{-2}$	7.6	$3.8 \cdot 10^{-3}$	54.6	$1.7 \cdot 10^{-3}$	18.5
8	$5.0 \cdot 10^{-1}$	—	$1.5 \cdot 10^{-2}$	4.7	$3.1 \cdot 10^{-3}$	12.2	$3.0 \cdot 10^{-4}$	12.5	$2.9 \cdot 10^{-5}$	57.8
16	$1.2 \cdot 10^{-1}$	4.2	$8.4 \cdot 10^{-4}$	17.4	$2.3 \cdot 10^{-4}$	13.4	$1.1 \cdot 10^{-5}$	27.7	$5.1 \cdot 10^{-7}$	56.8
32	$3.0 \cdot 10^{-2}$	4.0	$5.1 \cdot 10^{-5}$	16.4	$1.5 \cdot 10^{-5}$	15.5	$3.6 \cdot 10^{-7}$	30.6	$8.3 \cdot 10^{-9}$	62.0
64	$7.5 \cdot 10^{-3}$	4.0	$3.2 \cdot 10^{-6}$	16.0	$9.2 \cdot 10^{-7}$	16.0	$1.2 \cdot 10^{-8}$	31.1	$1.3 \cdot 10^{-10}$	64.0

$K$	$n = 6$	$R_C$	$n = 7$	$R_C$	$n = 8$	$R_C$	$n = 9$	$R_C$
2	$1.7 \cdot 10^{-2}$	—	$1.6 \cdot 10^{-3}$	—	$6.6 \cdot 10^{-4}$	—	$5.0 \cdot 10^{-5}$	—
4	$7.0 \cdot 10^{-5}$	245.2	$2.1 \cdot 10^{-5}$	77.3	$7.2 \cdot 10^{-7}$	909.9	$1.4 \cdot 10^{-7}$	355.4
8	$1.5 \cdot 10^{-6}$	46.9	$8.4 \cdot 10^{-8}$	248.1	$3.3 \cdot 10^{-9}$	221.1	$1.4 \cdot 10^{-10}$	961.4
16	$1.3 \cdot 10^{-8}$	119.5	$3.6 \cdot 10^{-10}$	230.7	$6.7 \cdot 10^{-12}$	486.8	$1.5 \cdot 10^{-13}$	957.3
32	$9.7 \cdot 10^{-11}$	128.9	$1.4 \cdot 10^{-12}$	254.2	$1.9 \cdot 10^{-14}$	360.8	$3.8 \cdot 10^{-15}$	40.1
64	$7.8 \cdot 10^{-13}$	124.9	$1.5 \cdot 10^{-14}$	94.6	$7.5 \cdot 10^{-15}$	—	$4.9 \cdot 10^{-15}$	—

Table 4: The 3D case. The errors in the mesh uniform norm and their ratios  $R_C$  in dependence on  $K = 2, 4, \dots, 64$  and  $n = \overline{1, 9}$  for algorithm (a)

solve it that is the nice result (especially taking into account the Matlab implementation of loops).

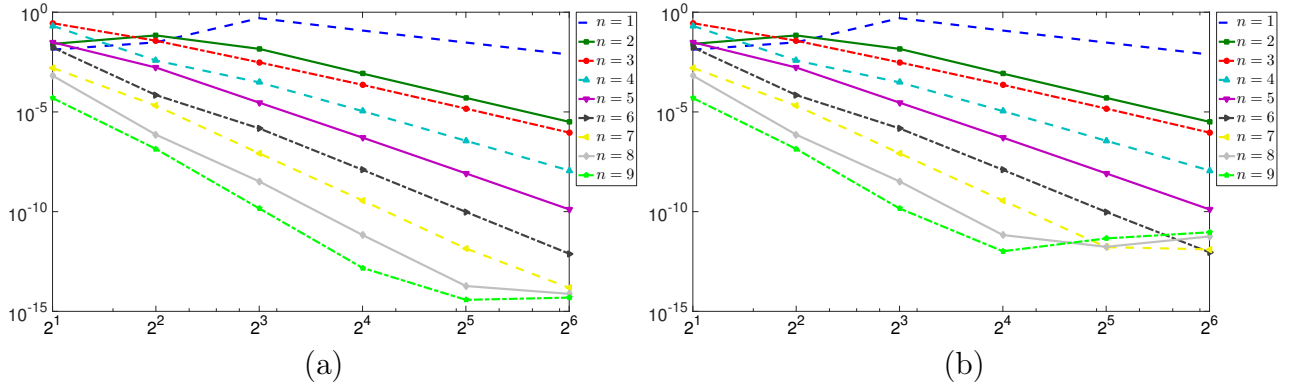


Figure 7: The 3D case. The errors in the mesh uniform norm for algorithms (a) and (b) in dependence on  $K = 2, 4, \dots, 64$  for  $n = \overline{1, 9}$

The study has been funded within the framework of the Academic Fund Program at the National Research University Higher School of Economics in 2016-2017 (grant no. 16-01-0054) and by the Russian Academic Excellence Project ‘5-100’ as well as by the RFBR, grant № 16-01-00048.

## References

- [1] Y.M. Altman, Accelerating MATLAB Performance: 1001 Tips to Speed up MATLAB Programs, Chapman and Hall/CRC, 2014.

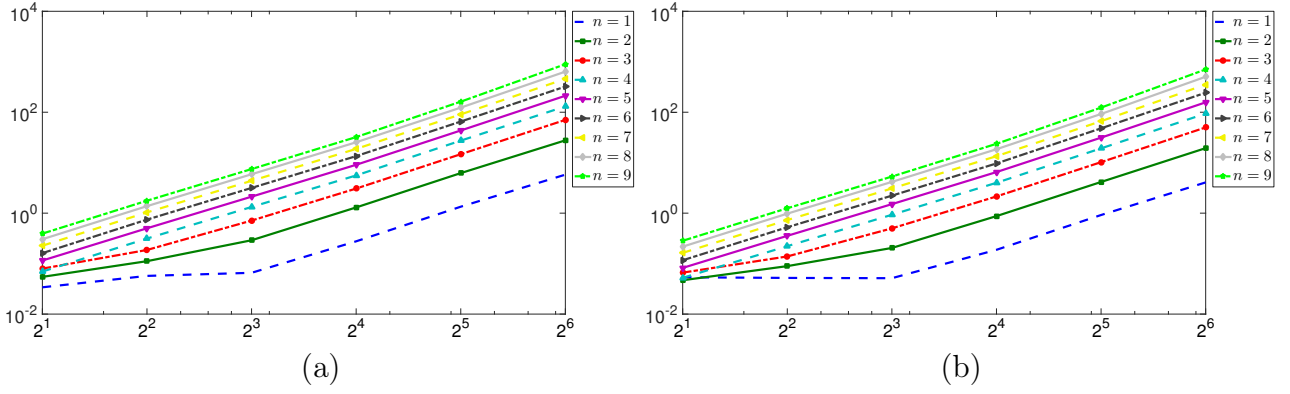


Figure 8: The 3D case. The execution time (in seconds) for algorithms (a) and (b) in dependence on  $K = 2, 4, \dots, 64$  for  $n = \overline{1, 9}$

$K$	$n = 1$	$n = 2$	$n = 3$	$n = 4$	$n = 5$	$n = 6$	$n = 7$	$n = 8$	$n = 9$
4	1.68	2.05	2.34	4.51	4.35	4.62	4.58	4.52	4.49
8	1.15	2.6	3.82	4.23	4.23	4.25	4.22	4.29	4.24
16	4.21	4.47	4.31	4.22	4.29	4.27	4.29	4.3	4.31
32	4.85	4.79	4.8	4.89	4.78	4.84	4.83	4.86	5.03
64	4.32	4.46	4.78	4.77	4.88	4.93	5.05	5.16	5.52

Table 5: The 3D case. The ratios of the execution times for algorithm (a) in dependence on  $K = 2, 4, \dots, 64$  and  $n = \overline{1, 9}$

$K$	$n = 1$	$n = 2$	$n = 3$	$n = 4$	$n = 5$	$n = 6$	$n = 7$	$n = 8$	$n = 9$
4	—	1.9	2.09	4.24	4.32	4.51	4.47	4.51	4.41
8	—	2.29	3.6	4.19	4.25	4.23	4.23	4.26	4.25
16	3.65	4.28	4.29	4.3	4.32	4.35	4.36	4.44	4.46
32	4.89	4.75	4.78	4.81	4.81	4.96	4.94	5	5.22
64	4.43	4.68	4.87	4.89	5.01	5.08	5.31	5.47	5.71

Table 6: The 3D case. The ratios of the execution times for algorithm (b) in dependence on  $K = 2, 4, \dots, 64$  and  $n = \overline{1, 9}$

- [2] B. Bialecki, G. Fairweather, A. Karageorghis, Matrix decomposition algorithms for elliptic boundary value problems: a survey, *Numer. Algor.* 56 (2011) 253–295.
- [3] V. Britanak, K.R. Rao, P. Yip, Discrete cosine and sine transforms: general properties, fast algorithms and integer approximations. Oxford: Academic Press – Elsevier, 2007.
- [4] P.G. Ciarlet, Finite Element Method for Elliptic Problems, SIAM, Philadelphia, 2002.
- [5] K. Du, G. Fairweather, Q.N. Nguyen, W. Sun, Matrix decomposition algorithms for the  $C^0$ -quadratic finite element Galerkin method, *BIT Numer. Math.* 49 (2009) 509–526.
- [6] B. Ducomet, A. Zlotnik, I. Zlotnik, The splitting in potential Crank-Nicolson scheme with discrete transparent boundary conditions for the Schrödinger equation on a semi-infinite strip, *ESAIM: Math. Model. Numer. Anal.*, 48:6 (2014), 1681-1699.
- [7] E.G. Dyakonov, Optimization in Solving Elliptic Problems, CRC Press, Boca Raton, 1996.
- [8] S. Eddins, Timing the FFT, <http://blogs.mathworks.com/steve/2014/04/07/timing-the-fft/>.
- [9] M. Frigo, S.G. Johnson, The design and implementation of FFTW3, *Proc. IEEE* 93 (2005) 216-231.
- [10] Y.-Y. Kwan, J. Shen, An efficient direct parallel spectral-element solver for separable elliptic problems, *J. Comput. Phys.* 225 (2007) 1721–1735.
- [11] A.A. Samarskii, E.S. Nikolaev, Numerical Methods for Grid Equations, Vol. I, Direct methods, Birkhäuser, 1989.
- [12] P.N. Swarztrauber, The methods of cyclic reduction, Fourier analysis and the FACR algorithm for the discrete solution of Poisson’s equation on a rectangle, *SIAM Review* 19:3 (1977) 490–501.
- [13] A. Zlotnik, I. Zlotnik, Finite element method with discrete transparent boundary conditions for the time-dependent 1D Schrödinger equation, *Kinetic Relat. Models* 5:3 (2012), 639–667.
- [14] A.A. Zlotnik, I.A. Zlotnik, Fast direct algorithm for implementation of the high order finite element on rectangles for boundary value problems for the Poisson equation, *Dokl. Math.* (2017) (in press).
- [15] <http://ltfat.sourceforge.net>